REMARKS

Applicants appreciate the continued thorough examination of the present application that is evidenced in the final Official Action of February 28, 2007 (the "Office Action"). Applicants respectfully request reconsideration of the rejections set forth therein for the reasons provided below.

Applicants appreciate the Examiner's careful consideration of Applicants' previously presented arguments, in which Applicants argued that U.S. Patent No. 6,141,705 to Anand et al. ("Anand") is directed to a system by which security processing is performed by an offload component (e.g. a NIC) under the direction and control of an application program as opposed to an operating system kernel. In response, the Office Action cites the following passage of Anand at col. 3, lines 9-23:

> In a preferred embodiment of the invention, a software implemented method and protocol is provided that allows, for instance, the operating system (OS) to "query" the device drivers (often referred to as "MAC" drivers) of any hardware peripherals (such as a NIC) that are connected to the computer system. The various device drivers each respond by identifying their respective hardware peripheral's processing capabilities, referred to herein as "task offload capabilities." In the preferred embodiment, once the task offload capabilities of each particular peripheral have been identified, the OS can then enable selected peripherals to perform certain tasks that could potentially be used by the OS. The OS can thereafter request that a peripheral perform the previously enabled task, or tasks, in a dynamic, as-needed basis, depending on the then current processing needs of the computer system.

Office Action at 9. Applicant submits that the foregoing passage of Anand, which is described by Anand as the "general inventive concept" (Anand, col. 1, l. 24), does not mean that in the system of Anand, an operating system kernel can initiate a request to a security offload component to secure a particular communication with a remote unit in response to receiving a request from an application program to initiate a communication with the remote unit, as recited in Claim 1. Unlike Claim 1, the actual control of the particular security processing to be performed is directed by a transport protocol driver, as described, for example, at col 10, ll. 27-63 of Anand. Nothing in the cited passage indicates that the operating system kernel can direct a

peripheral to secure a particular communication with a particular remote unit, much less to initiate operation of the security handshake processing by the security offload component, as recited in Claim 1.

Instead, the description of Anand makes it abundantly clear that security processing is performed according to requests from a transport protocol driver which, as Applicants previously explained, is not part of the operating system kernel. In response to this argument, the Office Action asserted that Anand discloses that the transport protocol driver is implemented in the operating system kernel. Office Action at 10. In support, the Office Action cited the following passage of Anand:

> In a preferred embodiment of the present invention, in the Windows NT layered networking architecture, a transport protocol driver, or transport, is implemented with an appropriate program method so as to be capable of querying each of the device driver(s) associated with the corresponding NIC(s) connected to the computer.

Anand, col. 3, ll. 45-50. Applicant has carefully studied the cited passage but respectfully disagrees with the examiner's conclusion. The cited passage describes the "Windows NT layered networking architecture," not the Windows NT operating system. Moreover, the passage states that the transport protocol driver "is implemented with an appropriate program method," and does not indicate that the driver is somehow implemented in the operating system kernel.

In contrast, Anand distinguishes between the operating system and the transport protocol driver. For example, as stated by Anand "[m]ore particularly, NDIS describes the interface by which one or multiple NIC drivers (116-120) communicate with one or multiple underlying NICs (100-104), one or multiple overlying transport protocol drivers, or transports, (represented at 128-134 in FIG. 2), and the operating system." (Emphasis added) Anand, col. 8, ll. 50-55. Thus, the transport protocol drivers are treated by Anand as distinct from the operating system. Anand goes on to say:

> All interactions between NIC driver and protocol driver, NIC driver and operating system, and NIC driver and NIC are executed via calls to wrapper functions. Thus,

instead of writing a transport-specific driver for Windows NT, <u>network vendors provide the NDIS interface as the uppermost layer of a single network driver</u>. Doing so allows any protocol driver to direct its network requests to the network card by calling this interface. Thus, a user can communicate over a TCP/IP network and a DLC (or an NWLINK, or DECnet, VINES, NetBEUI and so forth) network using one network card and a single network driver. (Emphasis added).

Anand, col. 8, l. 65 to col. 9, l. 8. Thus, the network driver, i.e. the transport protocol driver, is provided by the network vendor, and is clearly not part of the operating system kernel.

Having established that the network driver is not part of the operating system, Anand teaches that the security functions of a NIC 100 may be invoked <u>by the driver</u> by appending an appropriate packet extension to a data packet. See Anand, col. 10, ll. 27-47. In particular, Anand states:

For instance, in FIG. 3 the application data 140 is passed down from the upper layers of the network model to <u>an appropriate transport protocol driver</u>, such as TCP/IP 128. The <u>driver</u> repackages the data into an appropriate data packet 142. Then, depending on whatever additional functions are to be performed on this particular data packet 142 a functional component is included that appends a predefined data structure, referred to as the packet extension, to the data packet. (Emphasis added).
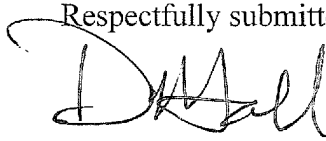
Anand, col. 10, ll. 35-38.

Accordingly, notwithstanding the "general inventive concept" cited in the Office Action, Anand expressly teaches away from an operating system kernel directing operations of a security offload component as recited in Claim 1.

Independent Claims 33-35 are submitted to be patentable for at least the same reasons as Claim 1. The dependent claims are patentable at least per the patentability of the independent claims from which they depend.

CONCLUSION

In light of the above remarks, Applicants respectfully submit that the above-entitled application is in condition for allowance. Favorable reconsideration of this application is respectfully requested. If, in the opinion of the Examiner, a telephonic conference would expedite the examination of this matter, the Examiner is invited to call the undersigned attorney at (919) 854-1400.

Respectfully submitted,

David C. Hall
Registration No. 38,904

Myers Bigel Sibley & Sajovec, P.A.
P.O. Box 37428
Raleigh, NC 27627
919-854-1400
919-854-1401 (Fax)